Kinsee: a Machine Learning Driven Home Workout App

Sera Vallee, Reyes Palomera Becerra, Huu Huynh, and Justin Norton

ABSTRACT Kinsee is a Machine Learning app that aims to extract keyframes from a workout, count the repetitions, and then evaluate one of three workouts using a Neural Network. Those workouts are sit-ups, pushups, and squats. We were able to achieve a 92% accuracy when trained on individuals, and 77% for a general model trained on n=4 individuals. Kinsee provides a proof of concept for an AI home-workout program. Using this method, we expect to be able to analyze a wide variety of exercises automatically. https://github.com/SRVallee/MLFitness_Capstone

INDEX TERMS Machine learning, motion capture, pose estimation, MediaPipe, Keras

I. INTRODUCTION

Over the COVID-19 pandemic there was a growth of home exercise, due to the closure of gyms. There is some evidence to suggest that the increase continues to persist even after most, if not all, of the pandemic restrictions have been lifted [1]. Additionally, there have been a number of recent developments in computer vision and machine learning involving the recognition and tracking of the human body. In conjunction with the explosion of home workout applications, we were interested in exploring integrating machine learning with home exercise in the form of an app: Kinsee.

a. Definitions and abbreviations

ML

Machine Learning

Repetition

Also known as a ‘rep’, is one complete exercise movement from start to finish,

Keyframe

Defines the starting and ending points of motion, particularly in animation or motion capture.

b. Literature Review

While there are some applications that already exist that already attempt to integrate exercise and machine learning, they tend to have two critical barriers to entry that we attempt to avoid. First, they may have physical trackers that need to be purchased separately in order to accurately track human movement (eg; Wei et al. [2]). Or, they may be inaccessible to the average person, requiring multi-camera set-ups or expert knowledge to interpret the data (eg; LinedanceAI [3]).

There have been a few projects that are similar to the one presented in this paper. [4] is an earlier example that only used machine learning to do rep counting, any pose evaluation is done by hard-coded checks to posture. [5] makes use of a random-forest classifier to determine the ‘terminal’ states of the exercise and thus determine if the pose is incorrect. however it requires the manual determination of the terminal state prior to training the model.

II. METHODOLOGY

The methodology described in this paper can be described as follows:

- ➢ Angle Computation
- ➢ Keyframe Extraction
- ➢ Rep Counting
- ➢ Input data into Machine Learning

For this project we were only looking to evaluate 3 different exercises due to time constraints, but leave room for further development into a greater number and more complex exercises. The exercises evaluated in this paper are sit-ups, push ups, and squats. We had access to 4 individuals to record videos.

a. Angle computation

In order to do the pose estimation, we decided to use MediaPipe's implementation of BlazePose (the version used in this paper is now the "legacy" version as of Apr 3, 2023)[6]. Through MediaPipe Pose we can predict 33 points, or landmarks, on the human body.

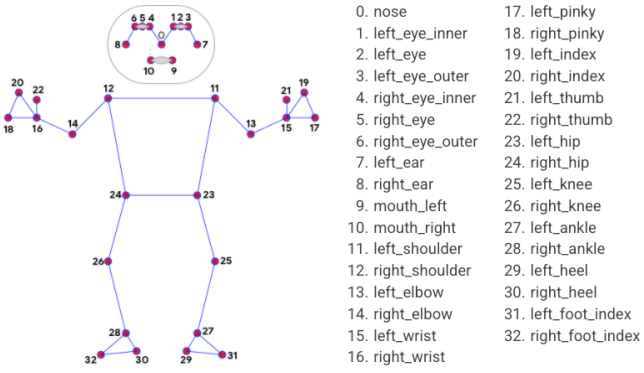


Fig 1. Pose landmarks [6]

However, as many of the landmarks are largely irrelevant to the application, we only track extract the 3D position of 14 of the total 33. From these points, we can compute the angle of each of the joints. For single-axis joints like elbows or knees, we need only compute the cosine angle between vectors through *eq 1*.

$$\cos\alpha = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\,\|\mathbf{B}\|}$$

where A and B are defined by taking the joint as the start point for each vector and the end points being the connected limb part. For the right elbow, vector A is vector from point 14 to 12 and B is 14 to 16 from Fig 1. For multiaxial joints like shoulders, we wish to extract the angles for flexion and extension; abduction and adduction; but not circumduction. We also use this computation to define angles around the spine, although not technically a multiaxial joint. To do so, we define three vectors V_1, V_2, K where K is the target vector, or the vector we wish to measure the angle of. To do so, we use *eq. 2*.

for $i = 1, 2$ where $\mathbf{n}_1 = \mathbf{V}_1 \times \mathbf{V}_2$ and $\mathbf{n}_2 = \mathbf{V}_1 \times \mathbf{n}_1$.

$$\cos\theta_i = \left(\mathbf{K} - \mathbf{n}_i \frac{\mathbf{K}\cdot\mathbf{n}_i}{\|\mathbf{n}_i\|^2}\right) \div \mathbf{V}_{3-i}$$

This projects the target vector onto an orthogonal plane to isolate and measure the angles discussed.

Body part	Angles computed
Head	2
Body	2
Shoulders	2 + 2
Hips	2 + 2
Elbows	1 + 1
Knees	1 + 1
Total	16

Table 1. Angles from each body part

BlazePose views the video at 30 frames per second and finds the landmark data for each frame. Thus, we have angle data for all 16 angles for each frame.

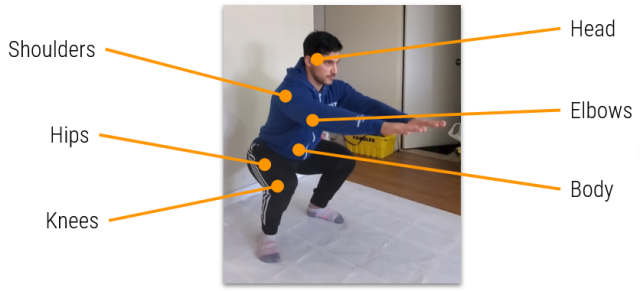


Fig 2. Angles tracked

b. Keyframe extraction

Since a video of 5 reps takes approximately 10 seconds, that results in 300 frames of data. However, the data is noisy and a function of time. To eliminate time from ML consideration and to reduce noise, we performed keyframe extraction on the data. We desired to select the most important frames to the total movement as our keyframes. From Xu et al.[7] we used their keyframe extraction method up until 3.1.2 including a modified version of Algorithm 1:

```
start, end=0
tempEnd=1
rSquared=x
while (start+1)!=n do
  while tempEnd<n do
    smallestR=R2(start, end)
    if smallestR >= rSquared then
      end=tempEnd
      tempEnd++
      if tempEnd!=n then
        for i in length(joints)
          curve=joints[i]

segments[i]=getBinomial(curve[start:temp+1],start)
    else
      keyList.append(frames[end], end)
      keyAngles.append(allAngles[end])
      start = tempEnd
      tempEnd++
      if start+1!=n then
        for i in length(joints)
          curve=joints[i]

segments[i]=getBinomial(curve[start:temp+1],start)
      break
```

The R^2 value is set by us to determine the sensitivity of the algorithm's keyframe extraction. A lower R^2 will result in less keyframes extracted and increasing it results in more keyframes extracted. For this project, we used an R^2 of 0.5 which reduces the number of frames by approximately 90%.

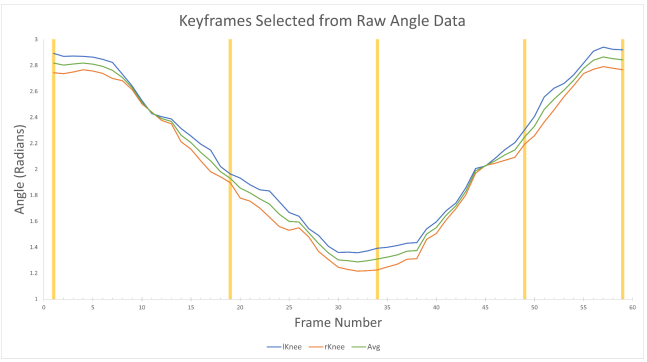


Fig 3. Keyframes extracted from angle data, represented by vertical yellow bars

In Fig 2, we can see that out of 58 frames, 5 frames were extracted.

c. Rep Counting

After extracting keyframes, we continue to extract repetitions from the workout. We start by declaring the important angles, and if the model does not exist yet, we separate them by getting angle cycles. We refer to a cycle as the angle of a joint closing and opening, which ends at a point where the angle stops changing. After getting the cycles, we ignore very small ones and add larger ones to the count. Each repetition stores the location in the video (frame number) for the start of the repetition, the turning point, and the end of the repetition. An example of a repetition can be seen in Fig. 2, where the first keyframe, third keyframe, and last keyframe are selected and the repetition itself. This method allows us to train any two-position exercise, such as push-ups, squats, sit-ups, etc., with just the initial setup of important angles.

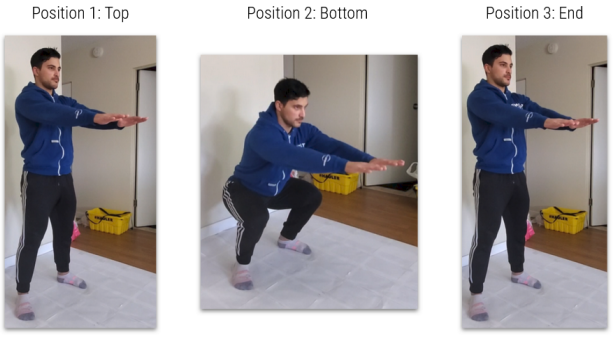


Fig 4. Examples of keyframes for one rep

d. Machine Learning

To perform the machine learning, we used the Keras API for Tensorflow[8]. The model is constructed by inputting known labeled videos (either correct or incorrect) for a specific exercise and feeding it into a neural network with two dense layers to construct a model for a particular exercise. The number of angles is dependent on the model, and may be up to 48; 16 angles times 3 keyframes. However, angles may be excluded if they are not relevant to the exercise.

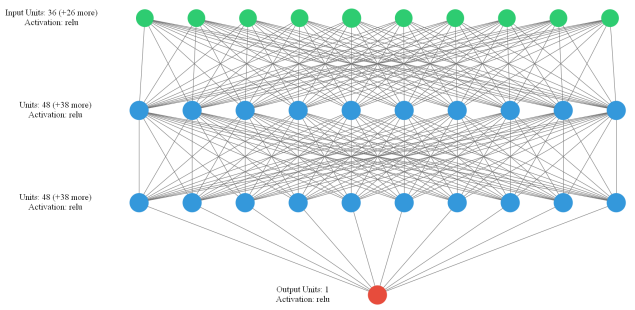


Fig 5. Neural Net with 36 input angles

III. RESULTS

We trained two separate models, an individual model, which has an approximate 92% accuracy over the 77% accuracy on the general model. This is because the individual model only needed to account for the same person and therefore did not need to account for the variation in body shape and size. Once the models have been trained, we can predict the correctness of the exercise.

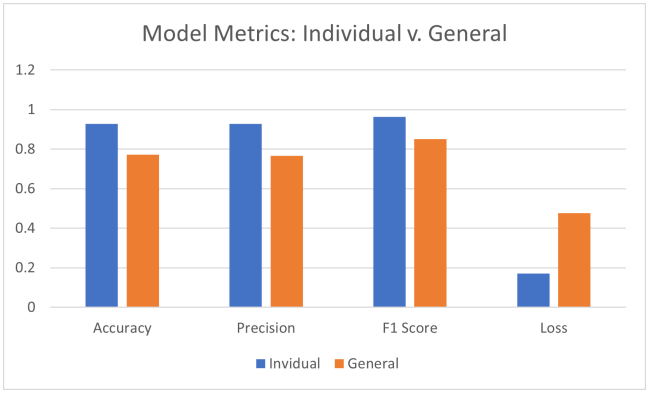


Fig 6. Model comparison for squats

To improve the performance of the machine learning models, we attempted to implement a dropout for the neural network to check for any biases in the data. However, the dropout never dropped any neurons from the network. Therefore, we removed it to make the program less heavy. We also attempted to train hyperparameters to increase the general model's accuracy; however, this caused the program to run for an unreasonable amount of time and so also had to be removed. We believe that this is due to insufficient parameters to account for the significant variation of an individual's body shape, size, camera angle, and distance. This is evident in the difference between Fig 7 and Fig 8. Fig 7 shows the variation of a specific angle at one of the 3 positions for one individual. Fig 8 demonstrates that the general model has to account for significantly more variation in these same angles between individuals.

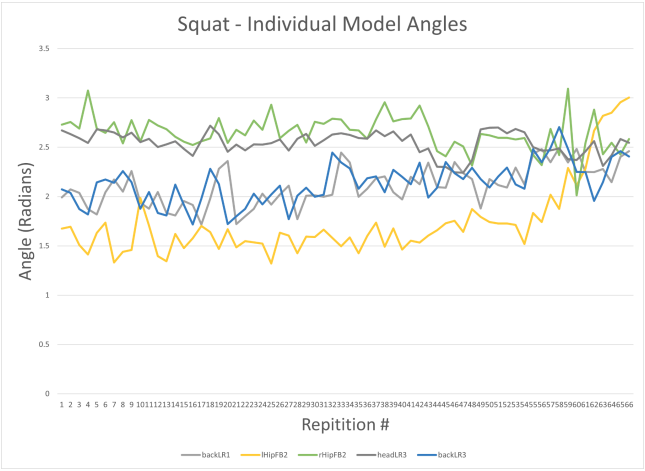


Fig 7. Select angles from individual squat model

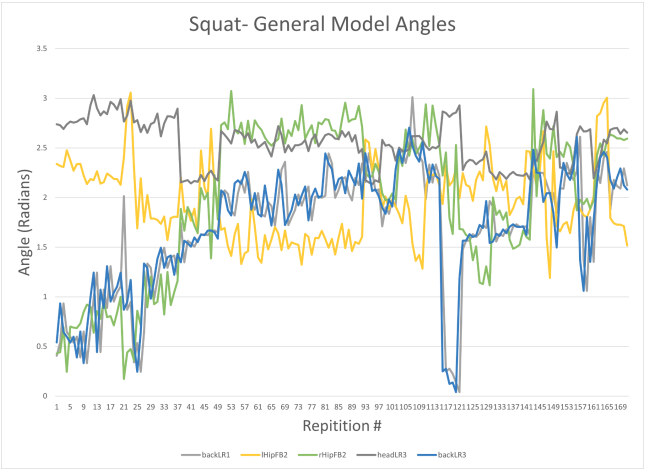
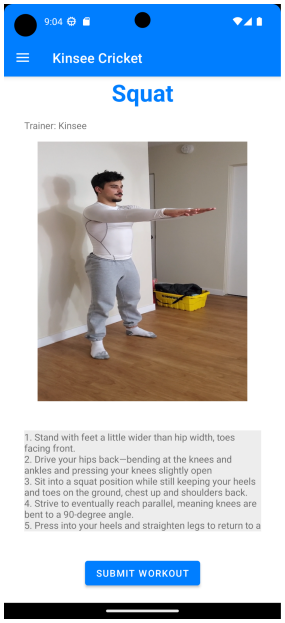


Fig 8. Select angles from general squat model

Users can access the general model through the evaluation feature in the Kinsee app. After logging in, the user can select the workout they wish to evaluate. This will bring up a demonstration video and instructions on how to do the workout. After pressing ‘submit workout’, the user will then be asked to record a video or upload a pre-recorded video.



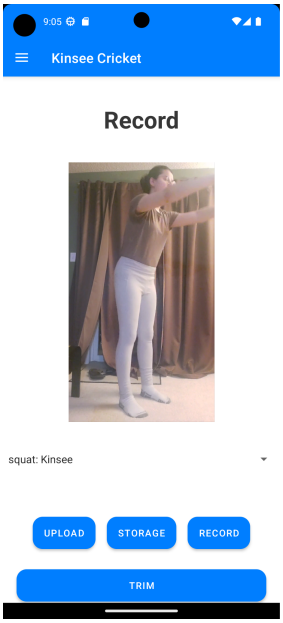


Fig 9. Exercise selection/demo screen
Fig 10. Record/upload screen

Aftering uploading the video, users must wait for the evaluation to finish on the server. Once complete, the video will appear in their ‘previous workouts’ on their homepage. They will be able to view the evaluated video along with the rep counting. There will also be feedback from the trainer, if the trainer has provided it.

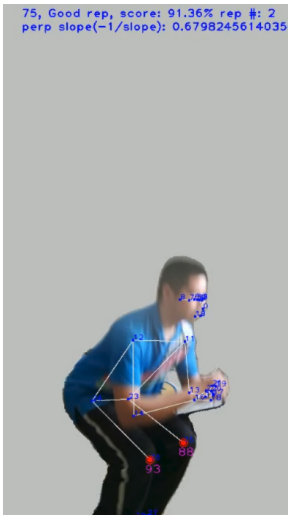


Fig 11. Still of an evaluated video

Each rep is scored individually, and the critical angles of workout are labeled. In this case, the knees.

IV. CONCLUSION

Overall, our exploration into this topic was successful. We can automatically detect keyframes and thus train a machine learning model based off of the extracted frames. From this model, a user can independently record and send videos to a server to evaluate them and provide the user with feedback using the models.

Our limitations were primarily associated with the machine learning aspect of this project. One of the largest issues was lack of sufficient raw data to build a genuinely robust model. As a result, the machine learning was less discerning than ideal. Another limitation as a result of the lack of data is that evaluated videos are severely restricted to a specific camera angle to be correctly evaluated. Additionally, the user can not currently access the individual model from the application or get specific feedback from the neural network.

In the future, we would like to expand the application functionality to improve usefulness and user experience. Specifically, to allow the user to build individual models and to enable the trainers to add and train their own workouts. To improve the machine learning performance, we would like to collect a significantly larger data set with which we can build the models.

Furthermore, we could move to a new machine learning method to give the user more specific workout feedback. Lastly, we would increase the number of keyframes that are used to analyze new workouts and improve the fidelity of the evaluations.

V. REFERENCES

[1] H. Shaban, “The pandemic’s home-workout revolution may be here to stay,” Washington Post, Jan. 07, 2021. Available: https://www.washingtonpost.com/road-to-recovery/2021/01/07/home-fitness-boom/

[2] W. Wei, K. Kurita, J. Kuang, and A. Gao, “Real-time limb motion tracking with a single IMU sensor for physical therapy exercises,” *Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* , Nov. 2021.

[3] LinedanceAI Movality, Available: https://www.linedanceai.com/ [Accessed Apr. 10, 2023].

[4] Haocong Ying, Tie Liu, Mingxin Ai, Jiali Ding, and Yuanyuan Shang. 2021. AICoacher: A System Framework for Online Realtime Workout Coach. In Proceedings of the 29th ACM International Conference on Multimedia (MM '21). Association for Computing Machinery, New York, NY, USA, 3787–3790. https://doi.org/10.1145/3474085.3478321

[5] V. Singh, A. Patade, G. Pawar and D. Hadsul, "trAIner - An AI Fitness Coach Solution," 2022 IEEE 7th International conference for Convergence in Technology (I2CT), Mumbai, India, 2022, pp. 1-4, doi: 10.1109/I2CT54291.2022.9824511.

[6] Google, “MediaPipe Pose,” *GitHub*, Apr. 5, 2023. [Online]. Available: https://github.com/google/mediapipe/blob/master/docs/solutions/pose.md. [Accessed: Apr. 10, 2023].

[7] Xu, C, Yu, W, Li, Y, Lu, X, Wang, M, Yang, X. KeyFrame extraction for human motion capture data via multiple binomial fitting. Comput Anim Virtual Worlds. 2021; 32:e1976. https://doi.org/10.1002/cav.1976

[8] Keras Team, “Keras API,” *Keras*. Available: https://keras.io/. [Accessed: Apr. 12, 2023].

A. APPENDIX

A1. Tables

Body part	Angles computed
Head	2
Body	2
Shoulders	2 + 2
Hips	2 + 2
Elbows	1 + 1
Knees	1 + 1
Total	16

Table 1. Angles from each body part

Metrics	Individual	General
Accuracy	0.92857	0.77142
Precision	0.92857	0.76667
F1 Score	0.96296	0.85185
Loss	0.17013	0.47604

Table 2. Model Comparison: Individual vs. General

A2. Algorithms

```
start, end=0
tempEnd=1
rSquared=x
while (start+1)!=n do
  while tempEnd<n do
    smallestR=R2(start, end)
    if smallestR >= rSquared then
      end=tempEnd
      tempEnd++
      if tempEnd!=n then
        for i in length(joints)
          curve=joints[i]
          segments[i]=getBinomial(curve[start:temp+1],start)
    else
      keyList.append(frames[end], end)
      keyAngles.append(allAngles[end])
      start = tempEnd
      tempEnd++
      if start+1!=n then
        for i in length(joints)
          curve=joints[i]
          segments[i]=getBinomial(curve[start:temp+1],start)
      break
```

A3. Figures

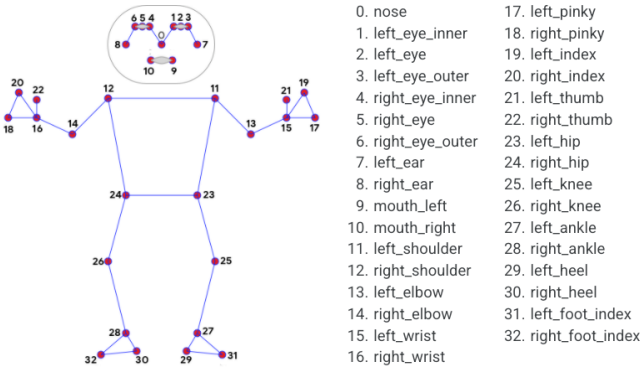


Fig 1. Pose landmarks [6]

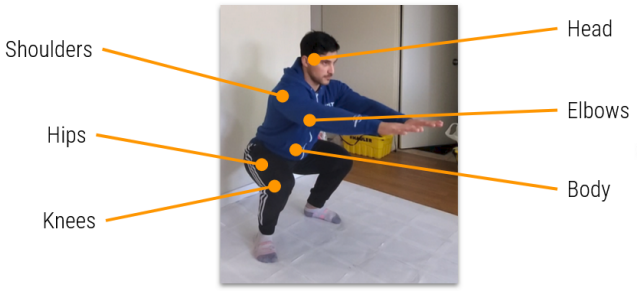


Fig 2. Angles tracked

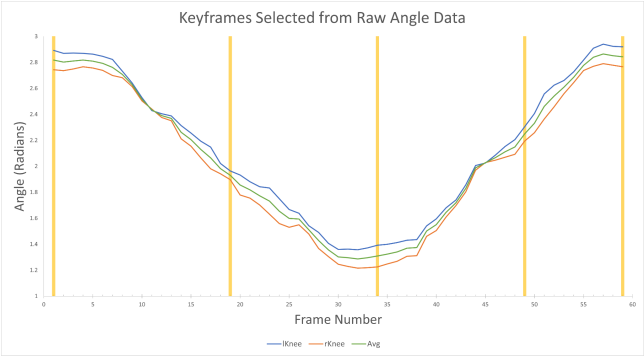


Fig 3. Keyframes extracted from angle data, represented by vertical yellow bars

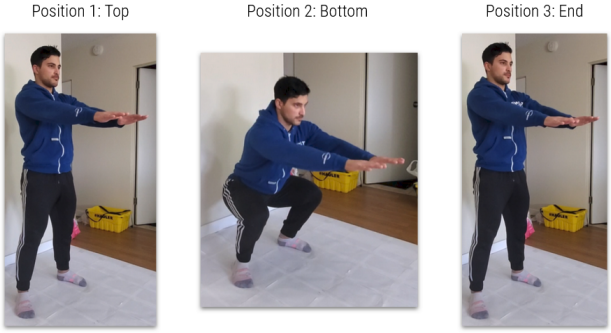


Fig 4. Examples of keyframes for one rep

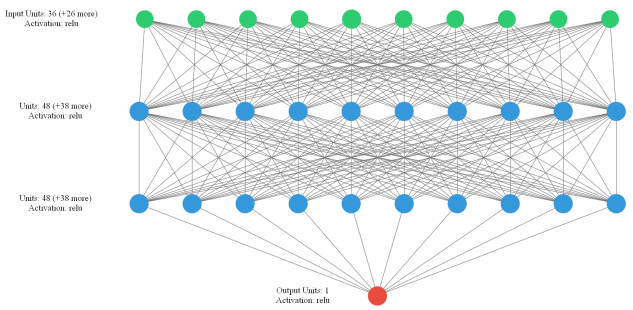


Fig 5. Neural Net with 36 input angles

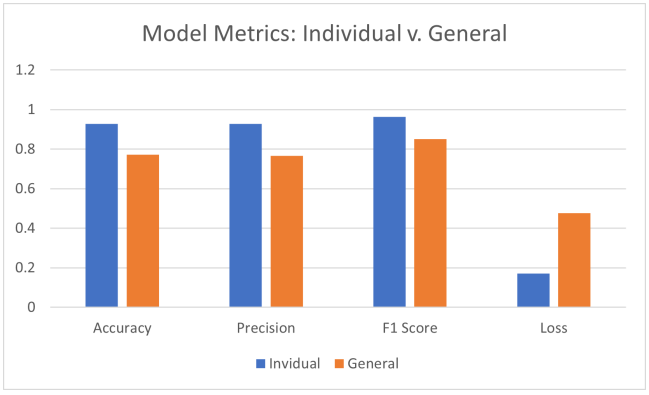


Fig 6. Model comparison for squats

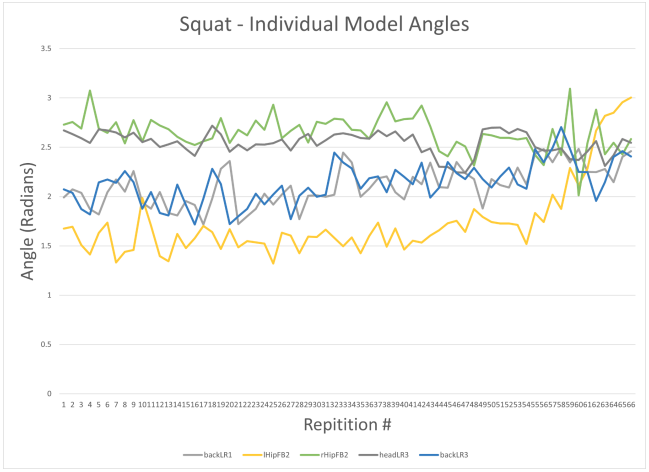


Fig 7. Select angles from individual squat model

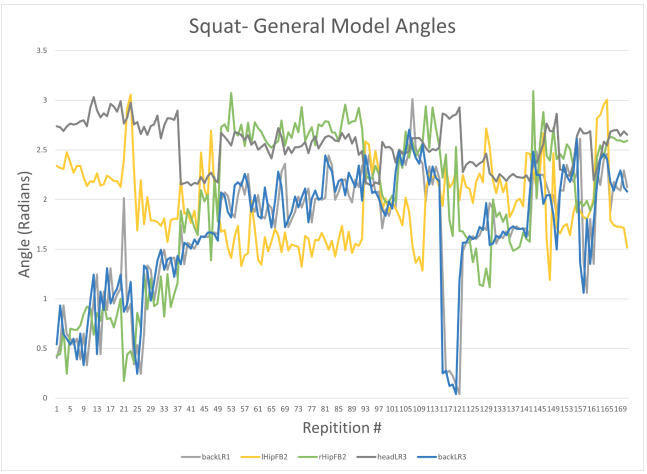


Fig 8. Select angles from general squat model

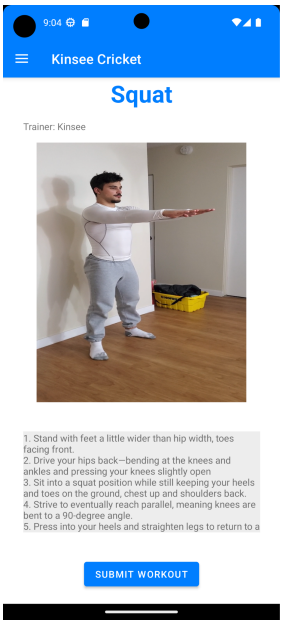


Fig 9. Exercise selection/demo screen

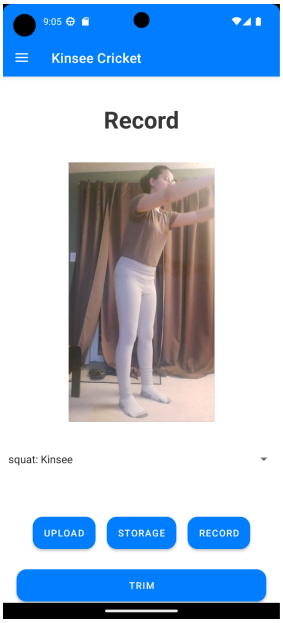


Fig 10. Record/upload screen

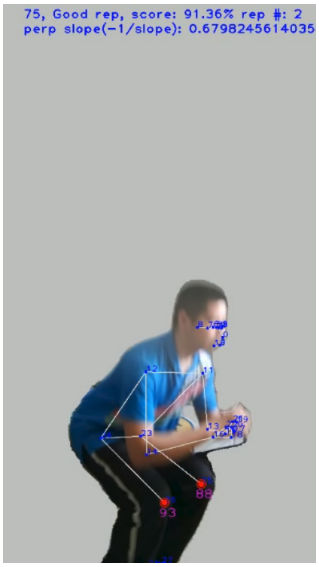


Fig 11. Still of an evaluated video